K. Pranitha
Data Analysis

1) Explain Programming and Python in detail.

Definition and purpose of Programming:-

Programming refers to a technological process for telling a computer which tasks to perform in order to solve problems.

Programming is the process of creating a set of instructions that tell a to instruct a computer to Perform specific tasks to solve problems or automate processes.

characteristics and applications of python:-

characteristics:-

⇒ Easy to Learn and use

⇒ Interpreted Language

⇒ High-level Language

⇒ Dynamically Typed

⇒ Objected-Oriented

⇒ Large standard Library.

⇒ Cross-Platform

Applications :-

⇒ web development

⇒ Data Science and Analytics

⇒ Artificial Intelligence

⇒ Machine Learning

⇒ Automation and Scripting

⇒ Software development

⇒ Game development

⇒ Internet of things

Types of comments in Python with syntax :-

Single-Line Comments :- Single line comments are used for brief explanations or to temporarily disable a single line of code.

Syntax :- The comment starts with a hash symbol (#) and continues to the end of the line. python ignores everything after the # on that line.

```
print("Hello, world!")
```

Multi-Line Comments :- Python does not have a dedicated syntax for multi-line comments like /* ... */ in other languages. Instead the methods uses.

⇒ Multiple Single line comments :- The conventional way is to use multiple consecutive lines, each starting with a hash Symbol.

⇒ print("Multi-line comments in python")

Triple-Quoted string Literals :- string literals enclosed in triple single quotes (''') or tiple double quotes(""") are ignored by the python interpreted.

⇒ Triple quoted string (''' or """) if they are not assigned to a variables.

⇒ print("string literals as comments")

Importance of Python in modern software development :-

Python holds a crucial role in modern software development due to its simplicity, versatility, and extensive ecosystem of libraries and frameworks.

⇒ Readability and Simplicity :- python's clean, English-like syntax emphasizes code readability, which makes it easier for developers to write, understand, and maintain code.

⇒ Versatility :- As a general-purpose language, Python is not restricted to a single domain.

⇒ Extensive Libraries and frameworks :- python boasts a rich ecosystem of third-party modules and a large standard library, providing pre-written code for almost any task.

2:- Describe Data types and operators in python wi
Suitable examples.

Built-in data types in python :-

Python has several built-in data types used to categorize and manage data. Everything in python is an object, and these data types are actually classes with variables being instances of these classes.

Numeric Types :-

=> Int :- Represents whole numbers (integers) without a decimal point. Integers in Python can be of unlimited length.

=> float :- Represents real numbers with a decimal point (floating-point numbers).

=> Complex :- Represents complex numbers with a real and an imaginary part, written with a "j" suffix.
e.g., 2+3j.

Sequence Types :-

=> List :- An ordered and mutable collection of items, enclosed in square brackets []. Lists are flexible and can contain items of different data types.

=> tuple :- An ordered and immutable collection of items, enclosed in parentheses ().

=> range :- A type used for generating sequences of numbers, often used in loops.

Mapping Type:-

=> dict (dictionary):- An unordered collection of data values used to store data in key-value pairs. Dictionary Keys must be unique and immutable.

Set type :-

=> Set :- An unordered and mutable collection of data unique items, defined using curly braces {} or set() function.

=> frozenset :- An immutable version of a set; elements cannot be changed after creation.

Boolean Type:-

=> bool :- Represents truth values, with only two possible values: True and False.

Type identification using type()

The built-in python type() function is used to determine the class type of an object at runtime. It is a fundamental tool in dynamically typed python for debugging, type checking and understanding the data.

Usage and Examples :- type() function has two primary uses:

=> Identifying the type of an object (single Arugament):- when called with a single argument, type() returns the exact type of that object, presented as a class

Syntax :- type(object)

=> Dynamically Creating a New class (Three Arguments):- In advanced use cases, type() acts as a metaclass (a template for a class) to create a new class

dynamically at runtime.

Syntax :- type(name, bases, dict)

Various Python Operators :-

Python operators are special symbols used to perform operations on variables and values. They are categorized into several groups. including.

⇒ arithmetic operators
⇒ assignment operators
⇒ comparison operators
⇒ Logical operators
⇒ Membership operators
⇒ Identity operators

Arithmetic operators :-

Arithmetic operators are used to perform mathematical calculations. They work with numbers like Integers and floats. use for mathematical calculations. Addition(+), Substraction(-), Multiplication(*), Division(/), Modulus(%), Floor division(//), Exponent(**).

Assignment operators :-

Assignment operators are used to assign or update values in a variables.

They combine arithmetic operation with assignment. used to assign values to variables.

Operators	Example
=	x = 5
+=	x += 3
-=	x -= 2
*=	x *= 2
/=	x /= 2

Comparison Operators :-

Comparison operators are used to compare two values. The result is always a boolean value (True or False). Used to compare two values. Result is True or False.

Operator	Meaning
==	Equal to
!=	Not equal
>	Greater than
<	less than
>=	Greator than or equal
<=	Less than or equal

Logical Operators :-

Logical operators are used to combine multiple conditions. They are commonly used in decision-making statements. Used to combine conditions.

Operators	Meaning
and	True if both are True
or	True if any one is True
not	Reverses the result.

Membership Operators :-

Membership operators check whether a value exists in a sequence. Used with lists, strings, tuples and sets. check whether a value exists in a sequence.

Operators	Meaning
in	present
not in	Not present

Identity Operators:-

Identity operators check whether two variables refer to the same memory object. They do not compare values, but object identity. check whether two variables refer to same object.

Operator	Meaning
is	same object
is not	Different object

Real-world usage of operators:-

⇒ Operators are fundamental symbols in programming and math that perform actions like calculations, comparisons, and logic with real-world uses in everything from calculating totals (arithmetic +, -) to controlling online discounts (logical AND, OR) and managing user permissions (bitwise).

---

3. Explain Python Input and Output operatoions in detail.

Input() function and its default data type:-

The Python input() function is used to get data from the user and it always returns the input as a string (str) data type by default.

The input() Function works:-

⇒ Functionality:- the program pauses and waits for the user to type something and press the Enter key.

⇒ Prompt :- An optional message (a string) can be included as an argument to guide the user.

⇒ Syntax :- variable_name = input("Optional prompt message")

Handling other Data types:-

input() returns a string by default, you must explicitly convert (typecast) the value if you need to work with other data types.

integers or floating-point numbers:-

For integers:- Use the int() function to convert the input string to an integer.

⇒ Syntax :- age = int(input("Enter your age: "))

For floating-point :- Use the float() function to convert the input string to a float.

⇒ Syntax :- height = float(input("Enter your height in meters: "))

Type conversion while taking input :-

When taking user input in most programming languages, the input is initially read as a string data type by default and must be explicitly converted to the desired data type.

Python is a common example where the input() function always returns a string.

⇒ int() :- converts the to an integer.

⇒ float() :- Converts to a floating-point number.

⇒ bool() :- Converts to a boolean

⇒ str() :- Converts to a string.

Taking multiple inputs:-

In Python, you can take multiple inputs in a single line using the split() method or list comprehension, or over multiple lines using a loop.

Taking Multiple Inputs in a Single line

This is the most common method for taking a fixed number of inputs separated by spaces on a single line

For strings:-

Ex:- first_name, last_name = input("Enter your first and last name: ").split()
print("First Name:", first_name)
print("Last Name:", last_name)

For numbers (integers or floats):

Taking Multiple Inputs into a List:-

if you need to handle a variable or large number of inputs, you can store them in a list.

using list comprehension:-

```
# user input: 1 5 2 8
numbers = [int(num) for num in input("Enter numbers
          Separated by spaces: ").split()]
print("Numbers list:", numbers)
```

Formatted Output using print(), Separators, and format Specifiers

Using the print() Function and the sep Parameter

The print() function can take multiple arguments, which it automatically Separates by a space by default.

→ Default behaviour (space separator):

```
print ("Name" ,"Age","city")
#Output : Name Age city
```

⇒ Custom separator (comma and space):

```
print ("Name", "Age", "city" , sep=" ,")
#output : Name, Age, city.
```

Using format Specifiers :-

You prefix the string with an f and use curly braces {} to embed variables and format Specifiers.

The basic syntax for a format Specifier inside curly braces {value : format_specifier}

Specifier	Description	Example
.2f	float with two decimal Places	print(f"Price {49.998:,2f}") #output : Price : 50.00
,	Thousands Separator	print(f"Population : {12345678:,}") #output : Population : 12,345,678
:^10	center align in a field of 10 chars	print(f"Name: {'Bob':^10}") #output: Name: Bob

4. Discuss Control Statements and Decision-Making statements in Python.

Meaning and importance of control statements:-

Control statements are essential programming constructs in Python that dictate the flow of execution within a program, enabling dynamic and logical behavior. Instead of code running sequentially line by line, control statements allow the program to make decisions and handle errors based on specific consditions.

Importance of control statements:-

Control statements are fundamental to writing efficient, readable and function are important for several reasons:

=> Decision Making
=> Code Efficiency and Reusability
=> Flexibility and Adaptability
=> Readability and structure.

Meaning of control statements:-

=> Conditional statement
=> looping statement
=> Control flow Altering statement
=> Exception Handling statements.

Types of control statements:-

In python, control statements are typically grouped into three main categories that manage the flow of Program execution.

=> Conditional Statements

⇒ looping statements

⇒ Jump statements.

Decision-making statements:-

Python provides several decision-making statements that allow a program to execute specific blocks of code based on whether certain conditions are True or false.

The primary decision-making statements are:-

⇒ if statement: The simplest form, which executes a code block only if a condition is True.

⇒ if-else statement: Provides an alternative block of code to run if the if condition is false.

⇒ if-elif-else statement: Allows for checking multiple conditions sequentially, executing the block associated with the first True condition.

Syntax flow and execution control with example:-

Syntax Flow:- Syntax flow means the order in which statements are written and executed in a program.

By default, a program runs from top to bottom. line by line

Example:-
```
print("start")
print("Learning python")
print("End")

#Output: start
        Learning python
        End.
```

Execution control:- Execution control decides which statement runs, when it runs, and how many times it runs.

It changes the normal top-to-bottom flow.

Three types :- conditional control, loop control, Jump control

⇒ conditional control :- used to execute code based on a condition.

Keywords :- if, elif, else.

Ex :-
```
age = 18
if age >= 18;
print("Eligible to vote")
else:
 print("Not eligible to vote")
```

Output :

Eligible to vote

loop control :- used to execute code multiple times

Key words : for, while

Ex :-
```
for i in range(3);
    print("Hello")
```

Output : Hello
Hello
Hello

5:- Write an essay on Python programming fundamentals.

Role of programming in problem solving :-

Its primary role is to provide a structured method and powerful tools for defining, designing, implementing, testing and optimizing solutions.

Key Roles of Python in problem solving :-

⇒ Problem Understanding and Analysis

⇒ Algorithm Design

⇒ Implementation

⇒ Testing and Debugging

⇒ optimization and Refinements.

Python Syntax simplicity and readability:-

Python's syntax is widely acclaimed for its simplicity and readability, a core design philosophy that makes it easy for both beginners and experienced developers to use.

Python avoids complex symbols and uses clear, short statements.

Features:-

=> English-like commands:- Python code looks close to normal english

Syntax:- print("Hello")

=> No semicolons or brackets:-

Syntax:-
```
a = 5
b = 10
print(a+b)
```

=> Indentation instead of curly braces

Syntax:-
```
if a>b:
    print("a is greater")
```

Readability:-

Python code looks clean and easy to understand even for beginners.

Features:-

=> Proper indentation improves structure

Syntax:- for i in range(3):

=> Simple and clear Keywords:-

uses English-like words such as if, else, for, while, print, which are easy to understand.

=> Identation instead of brackets:-

Blocks of code are defined by spaces/tabs, not {}

Syntax :- if age > 18 :
print ("Adult")

⇒ Easy variable declaration :-
No data type needed while declaring variables

* x = 10
name = "Sweety"

⇒ Readable function definitions :-

* add(a, b):
return a + b

Use of comments for code documentation :-
In python, comments are essential for enhancing code reabdability, providing context for developers, and facilitating maintenance and collaborates.

Primary Uses of Comments :-

⇒ clarification and context :- Comments explain non-obvious or complex logic, algorithms, or the reasoning behind specific coding decisions.

⇒ code Maintenance and collaborations :- They leave notes for future developers, making code easier to update, debug and maintain.

⇒ Planning and Reviewing :-
Comments can be used to write pseudocode or outlines before writing the actual source code, aiding in the planning process.

Programs:-

1. Movie Ticket Pricing

A movie theatre charges:
₹150 for children (age < 13)
₹250 for adults (age 13-59)
₹200 for Seniors (age ≥ 60)
If the person is watching a 3D movie, add ₹50 extra.
write a program that takes age and is3D (1 or 0) and prints the final ticket price.

```
age = int(input("Enter age :"))
is3D = int(input("Is 3D? (1=yes, 0=no): "))

if age < 13:
    price = 150
else:
    if age <= 59:
        price = 250
    else:
        price = 200
if is3D == 1:
    price = price + 50
print(price)
```

Output :-

```
Enter age : 18
Is 3D? (1=yes, 0=no) : 1
300
```

2. College Attendance Rule:

A student is allowed to write the exam if:
attendance ≥ 75
OR
attendance ≥ 60 AND has medical certificate (1=Yes, 0=no)
Take attendance percentage and medical certificate as input and print "Allowed" or "Not Allowed".

```
attendance = int(input("Enter attendance percentage:"))
medical = int(input("Has medical certificate? (1=Yes,0=
                                              no):"))
allowed = (attendance >= 75) or (attendance >= 60 and
                                  medical == 1)
if allowed:
    print("Allowed")
else:
    print("Not Allowed")
#Output:
Enter attendance percentage:66
Has medical certificate? (1=Yes,0=no):1
Allowed.
```

3) E-commerce Discount

A shopping site gives:

20% discount if bill ≥ 5000

10% discount if bill is between 2000 and 4999

No discount if bill < 2000

But if the customer is a prime member, they get extra 5% discount

Input: bill amount, isPrime (1 or 0)

Print final amount to be paid.

```
bill = float(input("Enter bill amount: "))
isPrime = int(input("Is prime member? (1=Yes,0=no):"))
discount = 0
if bill >= 5000:
    discount = 20
elif bill >= 2000:
    discount = 10
if isPrime == 1:
    discount += 5
final_amount = bill - (bill * discount /100)
```

```
print("Final amount to be paid:", final_amount)
#Output: Enter bill amount : 235678
        Is prime member? (1=yes,0=no): 0
        Final amount to be paid: 188542.4
```

4. Smartphone Battery warning

A phone shows:
"Low Battery" if battery ≤ 20
"Normal" if battery between 21-80
"Full" if battery > 80
But if phone is charging, it should show "charging" instead of any message.
Input: battery percentage, ischarging (1 or 0)

```
battery = int(input("Enter battery percentage:"))
ischarging = int(input("Is charging? (1=yes,0=no):"))
if ischarging == 1:
    print("charging")
else:
    if battery <= 20:
        print("Low Battery")
    elif battery <= 80:
        print("Normal")
    else:
        print("Full")
```

```
# Output:
Enter battery percentage: 80
Is charging? (1=yes,0=no): 0
Normal
```

5. (write an essay on python programming fundamentals. Explain)
Driving License check
A person can get a driving license if:
age ≥ 18
AND
passed driving test (1 = yes)

But if age ≥ 60, driving test is not required.

Input: age, testPassed

Print "Eligible" or "Not Eligible".

```
age = int(input("Enter age:"))
testPassed = int(input("Passed driving test? (1=Yes,0=no):"))
eligible = (age >=60) or (age >= 18 and testpassed == 1)
if eligible:
    print("Eligible")
else:
    print("Not Eligible")
```

```
#output:
Enter age: 20
Passed driving test? (1=Yes,
                      0=no):0
Not Eligible.
```

6. Online Food Delivery

A restaurant gives free delivery if:

Order amount ≥ 500

OR

User is a gold member

But if the distance is more than 10km, delivery is never free.

Input: amount, isGold (1 or 0), distance

Print "Free Delivery" or "Delivery charged".

```
amount = float(input("Enter order amount:"))
isGold = int(input("Is gold member? (1=Yes, 0=no):"))
distance = float(input("Enter distance (km):"))
free = (distance <=10) and (amount >= 500 or isGold == 1)
if free:
    print("Free Delivery")
else:
    print("Delivery charges Apply")
```

```
#output:
Enter order amount : 18
Is gold member? (1=Yes, 0=no) : 0
Enter distance(km): 22
Delivery charges Apply
```

Bank Loan Approval

A bank approves a loan if:

Salary ≥ 30,000 AND credit score ≥ 700

OR

Salary ≥ 50,000 (credit score ignored)

Input : salary, creditscore

Print "Loan Approved" or "Loan Rejected".

```
salary = int(input("Enter salary:"))
creditscore = int(input("Enter credit score:"))
approved = (salary >= 30000 and creditscore >= 700) or
                                (salary >= 50000)
if approved:
    print("Loan Approved")
else:
    print("Loan Rejected")
#output: Enter salary : 20000
         Enter credit score : 40000
         Loan Rejected
```

8. Electricity Bill

Units Consumed:

First 100 units → ₹ 2/unit

Next 100 units → ₹ 3/unit

Above 200 units → ₹ 5/unit

Note: No loops

print final bill amount

```
units = int(input("Enter units consumed:"))
bill = 0
if units <= 100:
    bill = units * 2
elif units <= 200:
    bill = (100 * 2) + (units - 100) * 3
else:
```

```
    bill = (100 * 2) + (100 * 3) + (units - 200) * 5
print(f"Total bill amount: ₹{bill}")
#Output: Enter units consumed: 200
         Total bill amount: ₹ 500
```

9: Student Scholarship

A student gets a scholarship if:

marks ≥ 85

AND

family income < 500000

But if the student is a single parent child, income condition is ignored.

```
Input: marks, income, singleParent (1 or 0)
marks = float(input("Enter marks:"))
income = float(input("Enter family income:"))
singleParent = int(input("Is single parent child? (1 for Yes, 0 for No):"))
if marks >= 85 and (income < 500000 or singleParent == 1):
    print("Scholarship Granted")
else:
    print("Scholarship Denied")
#Output: Enter marks: 40
         Enter family income: 15000
         Is single parent child? (1 for Yes, 0 for No): 1
         Scholarship Denied
```

10: Online Exam Result

A student passes if:

theory ≥ 40 AND practical ≥ 40

But if total (theory + practical) ≥ 100, pass even if one is less than 40.

Input :- theory, practical

```
theory = float(input("Enter theory marks : 60"))
practical = float(input("Enter practical marks :"))
if (theory >= 40 and practical >= 40) or (theory + practical
                                          >= 100):
    print("Result : Pass")
else:
    print("Result : Fail")
#output : Enter theory marks : 60
          Enter practical marks : 20
          Result : Fail
```

11:- Hotel Room Pricing

A hotel charges:

₹ 3000 per day for normal days

₹ 4000 per day on weekends

If customer stays more than 3 days, give 15% discount.

Input : isWeekend (1 or 0), daysStayed

print final bill.

```
isWeekend = int(input("Is it a weekend? (1=Yes, 0=NO):"))
daysStayed = int(input("Enter number of days stayed:"))
price_per_day = 4000 if isWeekend == 1 else 3000
total_bill = price_per_day * daysStayed
if daysStayed > 3:
    total_bill = total_bill * 0.85
print(f"Final bill: ₹{total_bill}")
#output :-
 Is it a weekend? (1=Yes, 0=NO): 1
```

Enter number of days stayed: 6
Final bill: ₹20400.0

12. Gaming Level unlock

A game unlock next level if:

Score ≥ 100

OR

player has a premium pass

But if player used cheating, access is denied.

Input: score, isPremium, usedcheat

```
Score = int(input("Enter score: "))
isPremium = int(input("Has premium pass? (1=Yes, 0=No): "))
usedcheat = int(input("Used cheating? (1=Yes, 0=No): "))
if usedcheat == 1:
    print("Access Denied")
elif score >= 100 or isPremium == 1:
    print("Level unlocked")
else:
    print("Level Locked")
#output: Enter score : 80
        Has premium pass? (1=Yes, 0=No): 1
        Used cheating? (1=Yes, 0=No): 0
        Level unlocked
```

13. Mobile Data Usage

A network gives unlimited data if:

daily usage ≤ 2GB

OR

User has unlimited plan

But if roaming is on, unlimited plan does not work.

Input: dataUsed, hasUnlimitedPlan, isRoaming

```
dataUsed = float(input("Data used (GB): "))
hasUnlimitedPlan = int(input("Unlimited plan (1/0): "))
isRoaming = int(input("Roaming (1/0): "))
if isRoaming == 1:
    print("Limited Data")
elif dataUsed <= 2 or hasUnlimitedPlan == 1:
    print("Unlimited Data")
else:
    print("Limited Data")
#output: Data used (GB): 2
        Unlimited plan (1/0) = 1
        Roaming (1/0): 0
        Unlimited Data
```

14:- Office Entry System
An employee can enter the office if:
ID card is valid
AND
(fingerprint matches OR face scan matches)
But if it is a holiday, entry is denied for everyone.
Input: idValid, fingerprint, faceScan, isHoliday

```
idValid = int(input("ID valid (1/0): "))
fingerprint = int(input("Fingerprint (1/0): "))
faceescan = int(input("Face scan (1/0): "))
isHoliday = int(input("Holiday (1/0): "))

if isHoliday == 1:
    print("Entry Denied")
elif idValid == 1 and (fingerprint == 1 or facescan == 1):
    print("Entry Allowed")
else:
    print("Entry Denied")
```

#output: ID valid (1/0): 1
Fingerprint (1/0): 0
Face scan (1/0): 1
Holiday (1/0): 0
Entry Allowed

---

15. Movie Rating Display
A movie app shows rating based on average score:
Average ≥ 8.5 → "Excellent"
Average between 6.0 and 8.4 → "Good"
Average < 6.0 → "Average"
But if the movie is marked as editor's choice, always show "Recommended".

Input: averageRating, isEditorschoice (1 or 0)
Print the message.

```
average = int(input("AverageRating :"))
isEditorschoice = int(input("Editors choice (1/0):"))
if isEditorschoice == 1:
    print("Recommended")
elif averageRating >= 8.5:
    print("Excellent")
elif averageRating >= 6.0:
    print("Good")
else:
    print("Average")
```

#output: Average Rating: 28
Editors choice (1/0): 0
Excellent